

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

OTONOM ARAÇLARLA
KAMPÜS ULAŞIM SİMÜLASYONU

Gonca Ezgi ÇAKIR

Danışman
Prof. Dr. Fatih Erdoğan SEVİLGİN

Haziran, 2021
Gebze, KOCAELİ
BIL496

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

OTONOM ARAÇLARLA
KAMPÜS ULAŞIM SİMÜLASYONU

Gonca Ezgi ÇAKIR

Danışman
Prof. Dr. Fatih Erdoğan SEVİLGİN

Haziran, 2021
Gebze, KOCAELİ
BIL496

Bu çalışma/...../2021 tarihinde ařağıdaki jüri tarafından Bilgisayar Mühendisliğı Bölümü'nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Prof. Dr. Fatih Erdoğan SEVİLGEN	
Üniversite	GEBZE TEKNİK ÜNİVERSİTESİ	
Fakülte	MÜHENDİSLİK FAKÜLTESİ	

Jüri Adı	Prof. Dr. Yusuf Sinan AKGÜL	
Üniversite	GEBZE TEKNİK ÜNİVERSİTESİ	
Fakülte	MÜHENDİSLİK FAKÜLTESİ	

ÖNSÖZ

Proje danışmanım olan Sayın Prof. Dr. Fatih Erdoğan SEVİLGİN'e ve Gebze Teknik Üniversitesi'nde eğitim aldığım tüm öğretim üyelerine emeklerinden dolayı teşekkürlerimi sunarım.

Ayrıca eğitimim süresince bana her konuda tam destek veren aileme saygı ve sevgilerimi sunarım.

Haziran, 2021

Gonca Ezgi ÇAKIR

İÇİNDEKİLER

ÖNSÖZ	I
ŞEKİL TABLOSU	IV
TABLO LİSTESİ	V
KISALTMA LİSTESİ	VI
ÖZET	1
SUMMARY	2
1. GİRİŞ	3
1.1. PROJENİN TANIMI	3
1.2. PROJENİN AMACI VE TASARIM PLANI	4
2. PROJE GEREKLİLİKLERİ	5
2.1. SİMÜLASYON GEREKLİLİKLERİ	5
2.2. OPTİMİZASYON PROBLEMİNİN TANIMLANMASI	6
2.3. SİSTEM, PROGRAMLAMA DİLİ VE KÜTÜPHANESİ	7
3. SİSTEM BİLEŞENLERİ	7
3.1. .XODR VE .FBX DOSYASI	8
3.2. CARLA İSTEMCİ – SUNUC BAĞLANTISI	8
3.3. DURAKLARIN KONUM BİLGİSİ LİSTESİ – TEXT DOSYASI	9
3.4. YOLCU BİLGİLERİ – TEXT DOSYASI	10
3.5. KAYNAK KOD DOSYALARI – PYTHON DOSYASI	11
4. YÖNTEM	12
4.1. KURULUM, HARİTA VE SİMÜLASYON	12
4.1.1. RoadRunner R2020b	12
4.1.2. CARLA Simülatör & Unreal Engine 4	15
4.2. AÇ GÖZLÜ ALGORİTMA	18
4.2.1. Rotadaki Araçların Yeni Rotaya İlerleme Kararı	18
4.2.2. Rotadaki Aracın Sıradaki Yolu Seçme Kararı	18

4.2.3.	Yeni Otonom Aracın Rotaya Başlaması	19
4.3.	KONTROL VE YOL NOKTASI KULLANIMI.....	19
4.4.	THREADLER	20
5.	SONUÇ.....	21
5.1.	SİMÜLASYON VE SÜRÜŞ SONUCU	21
5.2.	HARİTA TASARIM SONUCU	23
5.3.	PROJE GELİŞTİRİLİRKEN KARŞILAŞILAN ZORLUKLAR.....	24
6.	KAYNAKÇA	25

ŞEKİL TABLOSU

ŞEKİL 1.1	Proje Tasarım Şeması	5
ŞEKİL 2.1	Kütüphane Kullanımı İçin Kod Bloğu.....	7
ŞEKİL 3.1	CARLA Harita Aktarımı Şeması	8
ŞEKİL 3.2	İstemci – Sunucu (Client – Server) Bağlantısı Şeması	9
ŞEKİL 3.3	İstemci – Sunucu (Client – Server) Bağlantısı İçin Kod Bloğu.....	9
ŞEKİL 3.4	Durak Konum Bilgisi İçin .txt Dosyası	10
ŞEKİL 3.5	Yolcu Bilgisi İçin .txt Dosyası.....	11
ŞEKİL 4.1	RoadRunner ile Yol Eğiminin Belirlenmesi	13
ŞEKİL 4.2	RoadRunner Harita Tasarımı – Çevre Unsurları ve Duvar Varlıkları	14
ŞEKİL 4.3	RoadRunner Harita Tasarımı – Kuş Bakışı	14
ŞEKİL 4.4	RoadRunner Harita Tasarımı – Üç Boyutlu	15
ŞEKİL 4.5	RoadRunner ile OpenDrive Geçerlilik Kontrolü	15
ŞEKİL 4.6	CarlaUE4 Varlıklar – Marmaray Tren İstasyon	16
ŞEKİL 4.7	CarlaUE4 Varlıklar – Güvenlik Binası ve Kontrol Kapıları.....	17
ŞEKİL 4.8	CarlaUE4 Varlıklar – Basketbol Sahaları ve Hayvan Dostlarımız.....	17
ŞEKİL 4.9	CarlaUE4 Üzerinde Yol Noktalarının Gösterimi.....	19
ŞEKİL 4.10	Thread Senkronizasyonu – Sözde Kod.....	20
ŞEKİL 4.11	Yolcu Bilgisi ve Thread Senkronizasyonu ile Oluşturulması.....	21

TABLO LİSTESİ

TABLO 5.1 Aç Gözlü Yaklaşım Sonuçları	22
TABLO 5.2 Harita Tasarımının Kategorilere Göre Puanlama Sonuçları	23

KISALTMA LİSTESİ

GTÜ :	Gebze Teknik Üniversitesi
BIL495 :	GTÜ Bitirme Projesi 1 ders kodu
BIL496 :	GTÜ Bitirme Projesi 2 ders kodu
CarlaUE4 :	Carla Unreal Engine 4 Editor
ACO :	Ant Colony Optimization (Karıncı Kolonisi Optimizasyonu)
GRASP :	Greedy Randomized Adaptive Search Procedure (Aç Gözlü Uyarlanabilir Arama Prosedürü)

ÖZET

Teknolojinin sürekli geliştiđi bu dönemde, otonom araçlar hızla hayatımızda dahil olmuş ve önemli bir yer teşkil etmeye başlamıştır. Otonom araçlar gün geçtikçe geleceđin ulaştırma ve kentleştirmesini etkileyecek düzeyde ulaştırmanın temel unsurlarında biri haline gelmektedir. Otonom araçların kullanımıyla trafik güvenliđi, düzeni, verimi ve tasarruf hedeflenmektedir. Biz de projede otonom araçların insan ulaşımında kullanımına katkı sağlamayı hedefledik.

Günümüzde otonom araçların toplu taşıma, taksi, atık toplama gibi farklı amaçlarla kullanımları üzerine çalışılmaktadır. Bu çalışmalarda otonom aracın rotasını belirleyen algoritmalar üzerinde de durulmaktadır. Bu sayede ulaşımında verimlilik, araç, yakıt ve zaman tasarrufu sağlamak, hizmet kalitesini arttırabilmek ve çevre kirliliđine yol açan gaz salınımını azaltmak hedeflenmektedir.

Proje iki aşamadan oluşmaktadır, BIL495 kapsamında projenin ilk aşaması gerçekleştirilmiştir. Bu kapsamda GTÜ kampüsü içerisinde mevcut düzende yer alan; belirli bir rotada sabit kapasiteyle ulaşım sağlayan ring servisleri otonom araç olarak ele alındı. Üniversitemizin kampüs haritası tasarlanarak simülasyon ortamı hazırlandı ve otonom araç için duraklardaki yolcu talebine yönelik sabit bir rota belirlenerek sürüş gerçekleştirildi.

Projenin BIL496 kapsamında yer alan ve bu dönem yoğunlaşılın ikinci kısımda ise öncelikle kampüs haritası detaylandırılarak gerçeđe daha bir simülasyon ortamı sunmak hedeflendi. Rotanın dinamik olarak belirlenmesine otonom araç sayısı, otonom aracın kapasitesi, yolcu yoğunluđu ve konforunu göz önüne alınarak optimizasyon algoritmaları geliştirildi. Hesaplanan rota üzerindeki ulaşımın performans testi sayısal olarak tablolarla; görsel olarak simülasyon ortamıyla sunuldu.

SUMMARY

During this period, when technology is constantly developing, autonomous vehicles have quickly become involved in our lives and have begun to form an important place. Autonomous vehicles are becoming one of the key elements of transportation at a level that will affect the transportation and urbanization of the future. Traffic Safety, layout, efficiency and savings are targeted with the use of autonomous vehicles. We also aimed to contribute to the use of autonomous vehicles in human transportation in the project.

Currently, the use of autonomous vehicles for different purposes, such as public transport, taxi, waste collection, is being studied. In these studies, algorithms that determine the route of the vehicle are also focused on. In this way, it is aimed to save transportation efficiency, vehicles, fuel and time, to improve service quality and to reduce the gas release that leads to environmental pollution.

The project consists of two phases, the first phase of the project in the previous semester was carried out within the scope of BIL495. In this context, the ring located in the current layout within the GTU campus is considered as an autonomous vehicle that provides transportation with a certain capacity on a fixed route. A map of the campus was designed and a fixed route was set for passenger demand at the stops for the autonomous vehicle.

In the second phase of the project and the part that is focused on this semester, it is aimed to provide a more realistic simulation environment by detailing the campus map. Optimization algorithms have been developed by taking into account the number of autonomous vehicles, vehicle capacity, passenger density and comfort in route calculation. The performance test of transportation on the calculated route is presented numerically with tables; visually with a simulation environment.

1. GİRİŞ

Otonom teknolojileri günden güne daha gelişmekte ve daha karmaşık yapılar haline gelmektedir. Otonom araçların yakın gelecekte ulaştırma ve kentleştirmeyi etkileyecek düzeyde ulaştırmanın temel unsurlarında biri haline gelmesi beklenmektedir. Günümüzde otonom araçların toplu taşıma, taksi, atık toplama gibi farklı amaçlarla kullanımları üzerine çalışılmaktadır. Bu çalışmalarda araç rotasını belirleyen algoritmalar üzerinde de durulmaktadır. Bu sayede ulaşımında verimlilik, araç, yakıt ve zaman tasarrufu sağlamak; hizmet kalitesi arttırabilmek ve çevre kirliliğine yol açan gaz salınımını azaltmak hedeflenmektedir.

Hali hazırda piyasadaki araç üreticileri yüksek oranda otonom araçlar üreteceklerini duyurduklarından dolayı da otonom araçların yakın bir gelecekte gelişmiş ülkelerde görünürlüğünü artması beklenen bir durumdur. Dolayısıyla firmalar çalışmalarını hızla sürdürmektedir. [3] Bu kapsamdaki çalışmalarda bir otonom aracın ne olduğuna ve otonom sürüşteki seyir ve kontrol algoritmasının testi için planlanan aracın bir simülasyon modelinin gerçekleştirilmesine odaklanılmaktadır. CARLA Simülatör söz konusu durum için üretilmiş açık kaynaklı bir platformdur. Projede de bu platformdan yararlanılarak kendi belirlediğimiz ortamda sürüş gerçekleştiren otonom araçlar yer almaktadır.

Proje, BIL495 kapsamında geliştirilen projenin devam projesi olup, geliştirmede otonom aracın tasarımından rota algoritmasının optimizasyonuna odaklanılmıştır. Simülasyon ortamı için GTÜ kampüs ortamı seçilmiş ve tasarlanmış, bu alanda duraklar arası insan ulaşımının simülasyonu yapılmıştır.

1.1. PROJENİN TANIMI

Proje, iki aşamalı projenin ikinci aşamasıdır. Projenin ilk aşamasında kampüs ortamında yer alan otonom ring araçla insan ulaşımının simülasyonu gerçekleştirildi. Program girdisi olarak durakların konum bilgisi ve yolcu sayısı programa sunuldu. Veriler

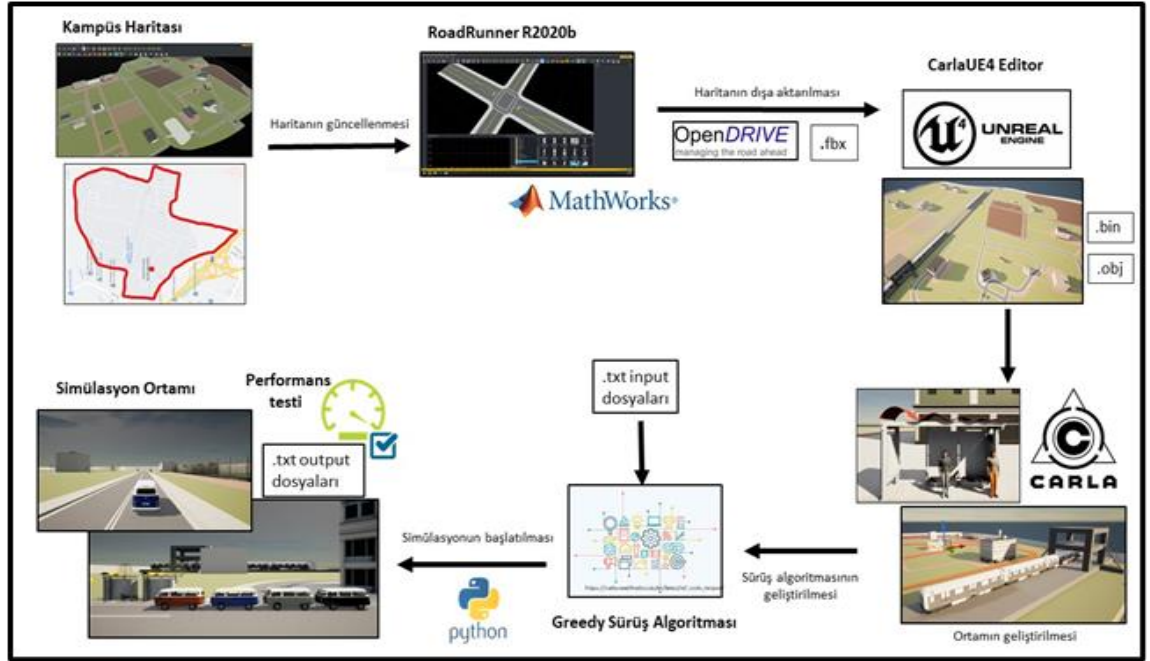
doğrultusunda belirtilen duraklarda yolcular oluşturuldu, basit bir algoritma ile rota belirlenerek tek bir aracın belirlenen rotada ilerlemesi sağlandı.

Projenin bu dönemki aşamasında ise, program girdisi detaylandırılarak yolcuların oluşma zamanları, bekledikleri ve varmak istedikleri durak bilgisi; durakların konum bilgisi programa sunuldu. Veriler doğrultusunda yolcular zamana göre dinamik olarak oluşturuldu. Rota algoritmasının optimizasyonu içinse “Aç Gözlü (Greedy) Algoritma” kullanıldı. Optimizasyon algoritmasının geliştirilmesinde otonom araç sayısı, araç kapasitesi ve yolcu talepleri parametre olarak belirlendi. Program çıktısı olarak her otonom aracın belirlenen rota üzerindeki sürüşü sayısal olarak taşıdığı yolcu sayısı ve rota mesafesi verileriyle tabloda; görsel olarak CARLA Simülatör ortamında sunuldu.

Simülasyonun gerçeği yansıtabilmesi için kampüsün haritası RoadRunner editör ve CarlaUE4 Editöre ile tasarlandı. Tasarımda kampüs ortamında yer alan duraklar, yeşil alanlar ve çevre unsurları eklenerek gerçeğe yakın bir simülasyon ortamı yaratıldı. Simülasyonda yer alan otonom aracın sürüşü yol noktası (waypoint) kullanımıyla gerçekleştirildi. Araç konumu kullanıcı isteğine göre kuş bakışı (2 boyutlu) ya da gözlemci (3 boyutlu) olarak harita üzerinde izlenebilir hale getirildi.

1.2. PROJENİN AMACI VE TASARIM PLANI

Bu projede, otonom araçlarla insan ulaşımını geliştirmek ve katkıda bulunmak amaçlanmıştır. Otonom araçlar için optimizasyon algoritmaları kullanılarak rota hesaplanması, hesaplanan rota üzerindeki duraklar arasında yolcu ulaşımının gerçekleştirilmesi ve performansın hem görsel hem sayısal olarak test edilmesi hedeflenmiştir. Proje için dönem başında planan akışı ŞEKİL 1.1 de görebilirsiniz.



ŞEKİL 1.1 Proje Tasarım Şeması

2. PROJE GEREKLİLİKLERİ

Bu bölümde projenin gerekliliklerinde bahsedilmiştir. Simülasyon gereklilikleri, optimizasyon probleminin tanımlanması, programlama dili ve kütüphanesi olmak üzere üçe ayrılmıştır.

2.1. SİMÜLASYON GEREKLİLİKLERİ

- Kampüs ortamının haritası güncellenecektir.
- Simülasyonda 4 araç yer alacaktır.
- Otonom araç kapasiteleri sabit olacaktır.
- Otonom araçlar sadece duraklarda duraklama yapacaktır.
- Yolcuların hangi zaman diliminde biniş ve iniş noktalarında bulunacağı belirlidir.
- Geliştirmede Greedy (Aç Gözlü) Algoritma ile meta sezgisel yöntemlerden GRASP (Greedy Randomized Adaptive Search Procedure), TS (Tabu Search), ACO (Ant Colony Opt) kullanılması hedeflenmektedir.

2.2. OPTİMİZASYON PROBLEMİNİN TANIMLANMASI

Enerji Verimliliği

Aracın rota boyunca kat ettiği mesafeye bağlı olarak tüketilen enerjinin göz önüne alınacak.

$$\text{Mesafe, } m(x) = \text{metre cinsinden alınan rotanın uzunluğu} \quad (2.1)$$

Sadece duraklarda duraklama olduğu göz önüne alınacak.

$$\text{Duraklar, } d(x) = \text{rotadaki durak sayısı} \quad (2.2)$$

Denklem 2.1 ve 2.2 ile enerji verimliliğini tanımlayabiliriz.

$$E(x) = m(x) + d(x) \quad (2.3)$$

Yolcu Konforu

Kapasite sınırına ulaşan araç yolcu alamayacağı için bazı yolcular yürümek zorunda kalacaktır.

$$\text{Konfor, } K(x) = \sum_{i=1}^{\text{durak sayısı}} \text{taşınamayan yolcu sayısı} \quad (2.4)$$

Amaç Fonksiyonu

Denklem 2.3 ve 2.4 ile amaç fonksiyonumuzu tanımlayabiliriz. Enerji verimliliği öncelik alındığından konfor denklemi α ($\alpha < 1$) değişkeni ile çarpılır.

$$\text{Min } f(x) = \sum_{i=1}^{\text{otonom araç sayısı}} E(x) + \alpha K(x) \quad (2.5)$$

2.3. SİSTEM, PROGRAMLAMA DİLİ VE KÜTÜPHANESİ

Projenin geliştirilmesinde yazılan programda kullanılan işletim sistemi, programlama dili, işletim sistemi ve kütüphane gereksinimleri şu şekildedir:

- Python 3.6.9
- Ubuntu 18.04 ve üzeri
- IDE olarak Visual Studio Code 2017 kullanılmıştır. Geliştirme için bu IDE' nin kullanılması zorunlu değildir. Başka bir IDE de kullanılabilir.
- Kütüphane olarak threadler için “time” ve “threading”, yönlü graf (directed graph) veri yapısı için “networkx”, simülasyon ortamı için ise CARLA'ya ait “carla” kütüphanesi kullanılmıştır. “carla” kütüphanesi içeri aktarıldığında (import) hata alınıyorsa simülatör kurulumunda gelen .egg dosyası kontrol edilmeli ve ŞEKİL 2.1 deki gibi dosya konumu doğru verilmelidir.

```
8
9  import glob
10 import os
11 import sys
12 import math
13 import random
14 from time import*
15 import threading
16
17 try:
18     sys.path.append(glob.glob('../carla/PythonAPI/carla/dist/carla-*%d.%d-%s.egg' % (
19         sys.version_info.major,
20         sys.version_info.minor,
21         'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
22 except IndexError:
23     pass
24
25 import carla
```

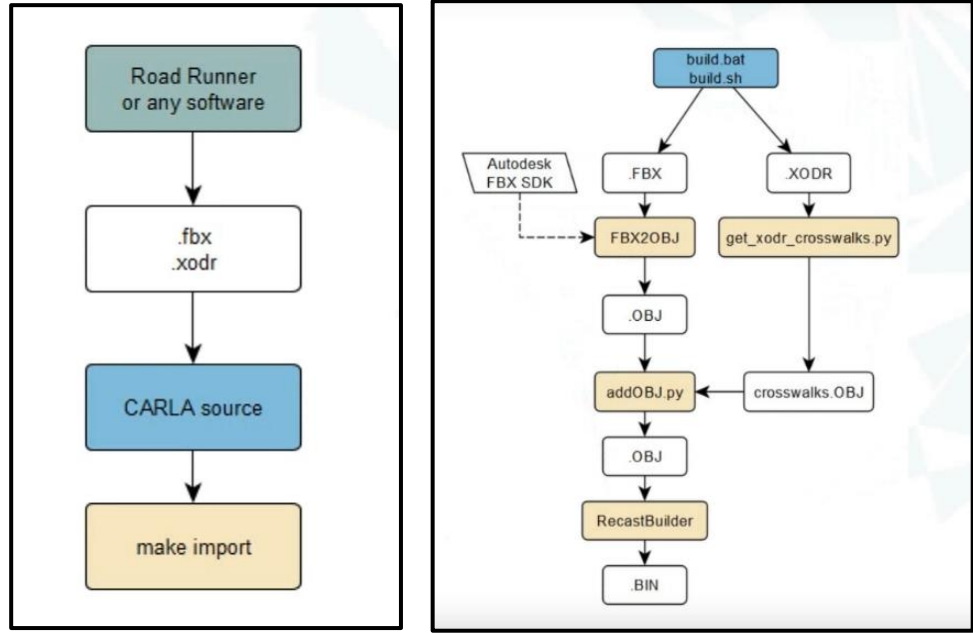
ŞEKİL 2.1 Kütüphane Kullanımı İçin Kod Bloğu

3. SİSTEM BİLEŞENLERİ

Bu kısımda tasarlanan haritanın simülasyon ortamına aktarılması, rota algoritmasının geliştirilmesi ve dinamik olarak yolcu oluşturmak için gereken dosyalar ve yapılar detaylı şekilde açıklanmıştır.

3.1. .XODR VE .FBX DOSYASI

.xodr ve .fbx uzantılı dosyaların içeri aktarılmasıyla (import) harita geometrisinin simülasyon ortamına aktarılması sağlanır. CarlaUE4 Editör' üne ait Carla Exporter kullanarak harita geometrisi .obj uzantılı dosyaya aktarılır. Son olarak .xodr (OpenDrive) ve .obj uzantılı dosyalar RecastBuilder ile .bin uzantılı yaya navigasyon bilgisini tutan dosyanın oluşturulması sağlanır. (ŞEKİL 3.1) Bu şekilde tasarlanan harita başarı şekilde simülasyon ortamına aktarılır, belirtilen konumlarda yolcular oluşturulabilir.



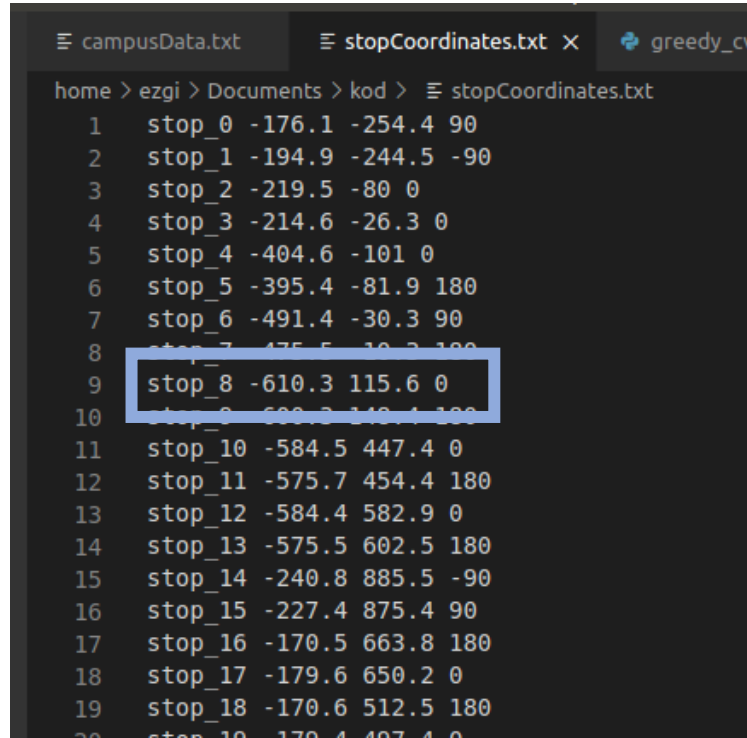
ŞEKİL 3.1 CARLA Harita Aktarımı Şeması

3.2. CARLA İSTEMCİ – SUNUC BAĞLANTISI

Simülasyon gerçekleştirilmeden önce simülatör için istemci – sunucu (server – client) bağlantısı sağlanmalıdır. Bu işlem kodda client için gerekli port bilgileri verilip (ŞEKİL 3.3), CarlaUE4 editöründeki “Play” butonuyla server bağlantısı kurularak gerçekleştirilir. Bağlantı sağlanmadığı takdirde alınan girdi dosyası doğrultusundan elde

konumuna yönelik x ve y koordinat bilgisi belirtilmiştir. z koordinatı yüzey üzerinde yer aldığı için sabit olarak 1.85 olarak belirlenmiştir, dosyada yer almamaktadır.

- Her durak harita üzerindeki merkez konuma göre belli bir açıda konumlandırılmıştır. 0, 90, 180 ve -90 olmak üzere merkez noktasına göre açılar da her durak için belirtilmiştir.



```
home > ezgi > Documents > kod > stopCoordinates.txt
1 stop_0 -176.1 -254.4 90
2 stop_1 -194.9 -244.5 -90
3 stop_2 -219.5 -80 0
4 stop_3 -214.6 -26.3 0
5 stop_4 -404.6 -101 0
6 stop_5 -395.4 -81.9 180
7 stop_6 -491.4 -30.3 90
8 stop_7 -475.5 -30.3 180
9 stop_8 -610.3 115.6 0
10 stop_9 -588.3 148.4 180
11 stop_10 -584.5 447.4 0
12 stop_11 -575.7 454.4 180
13 stop_12 -584.4 582.9 0
14 stop_13 -575.5 602.5 180
15 stop_14 -240.8 885.5 -90
16 stop_15 -227.4 875.4 90
17 stop_16 -170.5 663.8 180
18 stop_17 -179.6 650.2 0
19 stop_18 -170.6 512.5 180
20 stop_19 -170.4 407.4 0
```

ŞEKİL 3.4 Durak Konum Bilgisi İçin .txt Dosyası

3.4. YOLCU BİLGİLERİ – TEXT DOSYASI

Aşağıda yer alan “campusData.txt” girdi dosyasında görüldüğü gibi, sırasıyla her yolcu için yolcu numarası, beklediği durak adı, oluşma zamanı, varmak istediği durak adı, ortamdan yok olma zamanını belirtilmiştir. (ŞEKİL 3.5)

- Her yolcunun numarası (ID) listedeki sırasına bağlı olarak belirtilmiştir.

- Yolcunun bekleme zamanı ve ortamdan yok olma zamanı pozitif tam sayı olarak belirtilmiş. Bu sayılar simülasyon süresince threadler ile devamlı sayılan sayaçtaki değerlere eşit olduğunda gerekli durakta oluşması ya da ortamdan yok olması için gereklidir.
- Yolcunun beklediği durak ve inmek istediği durak ise “stopCoordinates.txt” dosyasındaki isimlerle aynıdır. Gerekli durak ismi aynı formatta belirtilmiştir.

```

campusData.txt
home > ezgi > Documents > kod > campusData.txt
1 4
2 1 stop_101 1 stop_28 15
3 2 stop_101 1 stop_28 15
4 3 stop_28 2 stop_101 15
5 4 stop_28 2 stop_101 15
6 5 stop_28 2 stop_101 15
7 6 stop_28 2 stop_101 15
8 7 stop_0 3 stop_101 15
9 8 stop_0 3 stop_101 15
10 9 stop_0 3 stop_101 15
11 10 stop_2 3 stop_101 15
12 11 stop_2 3 stop_101 15
13 12 stop_2 3 stop_101 15
14 13 stop_2 3 stop_101 15
15 14 stop_2 3 stop_101 15
16 15 stop_4 3 stop_101 15
17 16 stop_4 3 stop_101 15
18 17 stop_4 3 stop_101 15
19 18 stop_2 3 stop_101 15
20 19 stop_2 3 stop_101 15

```

ŞEKİL 3.5 Yolcu Bilgisi İçin .txt Dosyası

3.5. KAYNAK KOD DOSYALARI – PYTHON DOSYASI

Program 4 adet .py uzantılı kaynak kod dosyası bekler.

- “vehicle.py” otonom araca ait değişkenleri tutan bir sınıf yapısı barındırır. Sınıf içerisinde araç numarası, araç kapasitesi, duraklama yapılan durak sayısı, taşınan yolcu sayısı, carla araç objesi, carla araç objesi transformation bilgisi için değişkenler bulunur.

- “passenger.py” ise yolcuya ait değişkenleri tutan bir sınıf yapısı barındırır. Sınıf içerisinde yolcu numarası, beklediği durak, oluşma zamanı, varmak istediği durak, bekleme süresi, bekleme sınırı, carla yolcu objesi , carla yolcu objesi türü ve transformation bilgisi için değişkenler bulunur. Aynı zamanda yolcuların simülasyon ortamında oluşturulabilmesi için gerekli fonksiyonu da barındırır.
- “diGraph.py” ise “networkx” kütüphanesine ait DiGraph() fonksiyonunu kullanarak yönlü graf (directional graph) veri yapısı oluşturur. Kampüs haritasında yer alan durakları vertex (köşe), duraklar arasındaki yolları edge (kenar), ve yol uzunluklarını edge weight (kenar ağırlığı) olarak graph yapısına aktaran fonksiyonu içerir.
- “greedy_cvrp.py” ise ana kaynak koddur; program bu dosya üzerinden çalışır. CARLA simülasyon ortamına ait çevresel koşulların oluşturulması, gerekli girdi dosyalarının okunup saklanması, otonom aracın sürüşü ve greedy rota algoritmasının gerçekleştirilmesi için gerekli fonksiyonları barındırır. Aynı zamanda zamana bağlı yolcu oluşturmak için thread kullanır.

4. YÖNTEM

Bu kısımda CARLA Simülatör’de otonom araç için simülasyon ortamını ve rota algoritması geliştirmekte uygulanan yöntemlerin açıklaması yapılmıştır.

4.1. KURULUM, HARİTA VE SİMÜLASYON

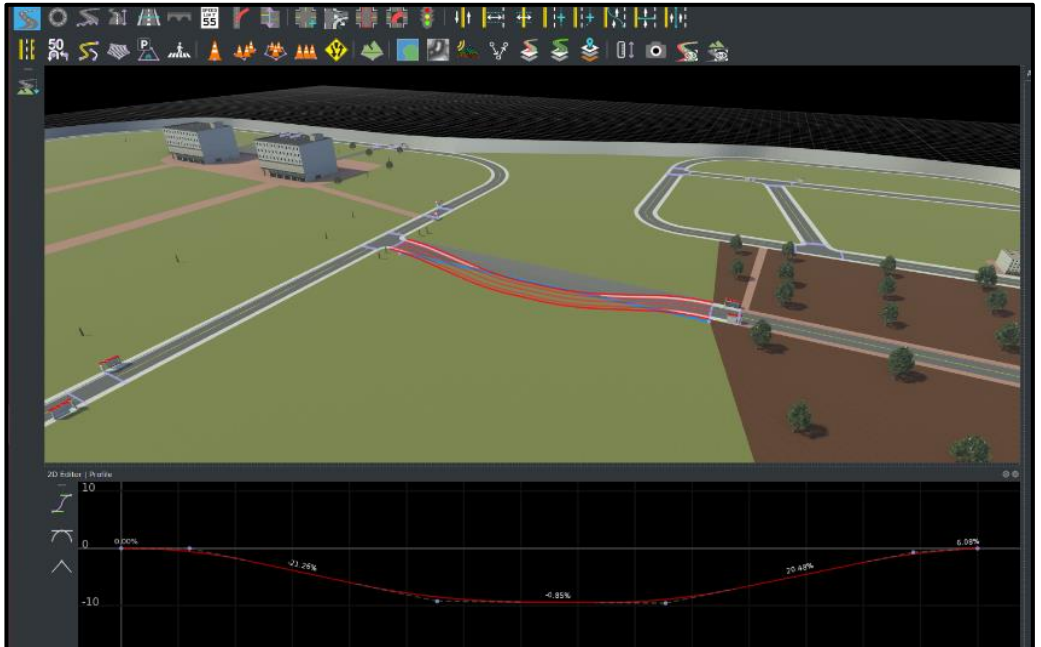
BIL495 projesinde yaşanan donanım sorunlarını yaşamamak adına yeterli donanıma sahip bir bilgisayar kiralandı. Bu bilgisayar üzerinde Ubuntu 18.04 işletim sistemi kuruldu. Sonrasında simülasyon ortamını tasarlamak için CARLA Simülatör ve lisansı okul tarafından sağlanan RoadRunner R2020b kurulumu yapıldı.

4.1.1. RoadRunner R2020b

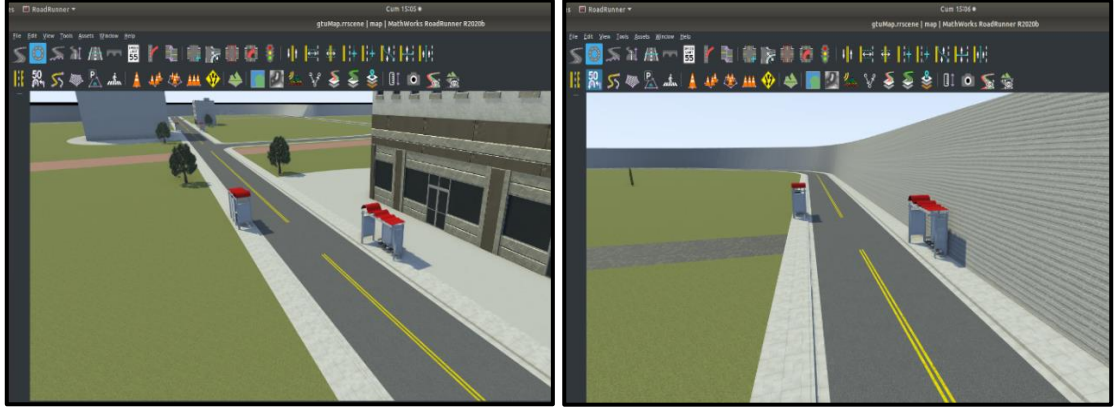
RoadRunner, CARLA Simülatör ile uyumlu olarak çalışabilen, otomatik sürüş sistemlerini simüle etmek ve test etmek için 3D sahneler tasarlamayı sağlayan

etkileşimli bir editördür. [4] Mathworks' e ait lisanslı bir yazılımdır ve okulun kampüs genelinde sağladığı Mathworks lisansı sayesinde iki dönem boyunca proje geliştirilirken ücretsiz olarak faydalanılmıştır.

BIL495 kapsamında RoadRunner R2020a ile geliştirilen haritadan da faydalanılarak RoadRunner R2020b üzerinde yeni ve daha detaylı bir harita tasarımı yapıldı. Geliştirme sırasında Google Harita verileri ve GTÜ web sitesinde yer alan kampüsün basit krokisi de göz önüne alındı. Harita görsellerine dayanarak yollar yeniden konumlandırıldı ve dönüş/ kesişim bulunan yollar için bağlantı noktaları eklendi. Yokuş ve alt geçit olarak kullanılan yollarda eğim değerleri ayarlanarak (ŞEKİL 4.1) yüzey seviyesinin altında yer alması sağlandı. Kaldırım ve yürüyüş alanları, ormanlık araziler ve yeşil alanlar editördeki araç (tool) yardımıyla çizilerek ortama eklendi. Binalar, duraklar, ağaçlar ve bitkiler RoadRunner' a ait hazır varlıklardan seçilerek haritada konumlandırıldı. Son olarak simülasyon ortamında daha net bir sonuç elde etmek amacıyla haritanın dış çevresi duvar varlıklarıyla çevrelendi. (ŞEKİL 4.2)

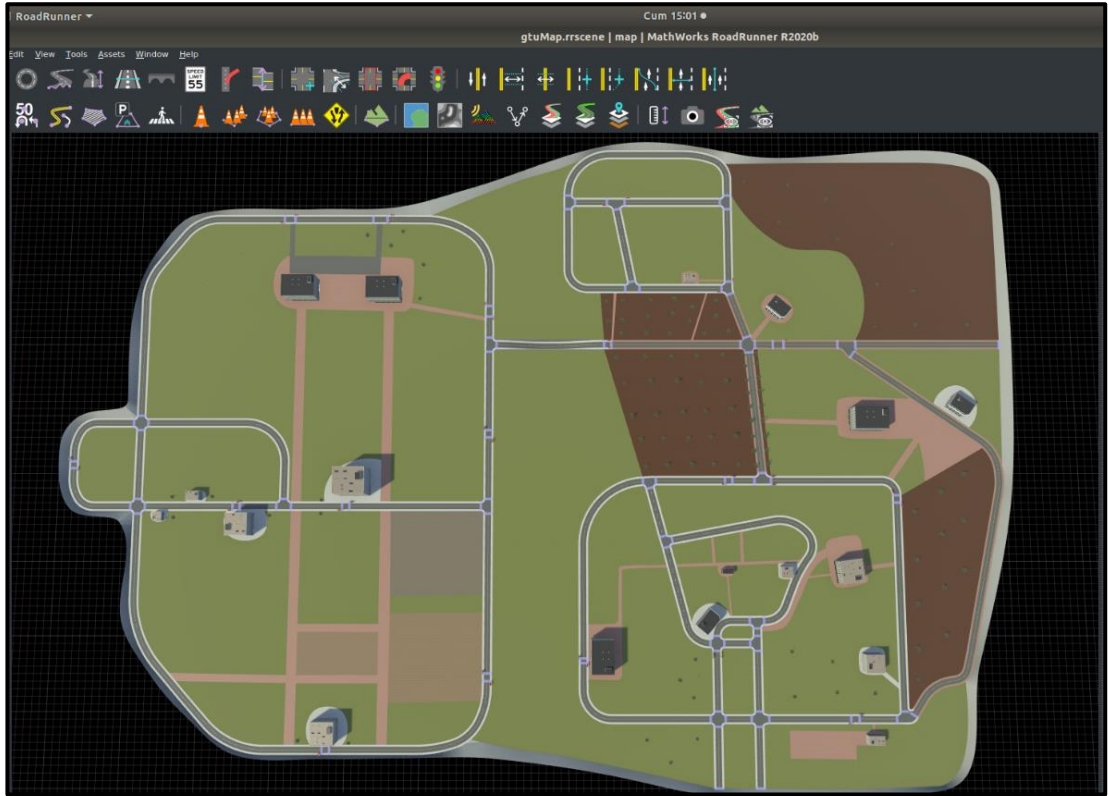


ŞEKİL 4.1 RoadRunner ile Yol Eğiminin Belirlenmesi

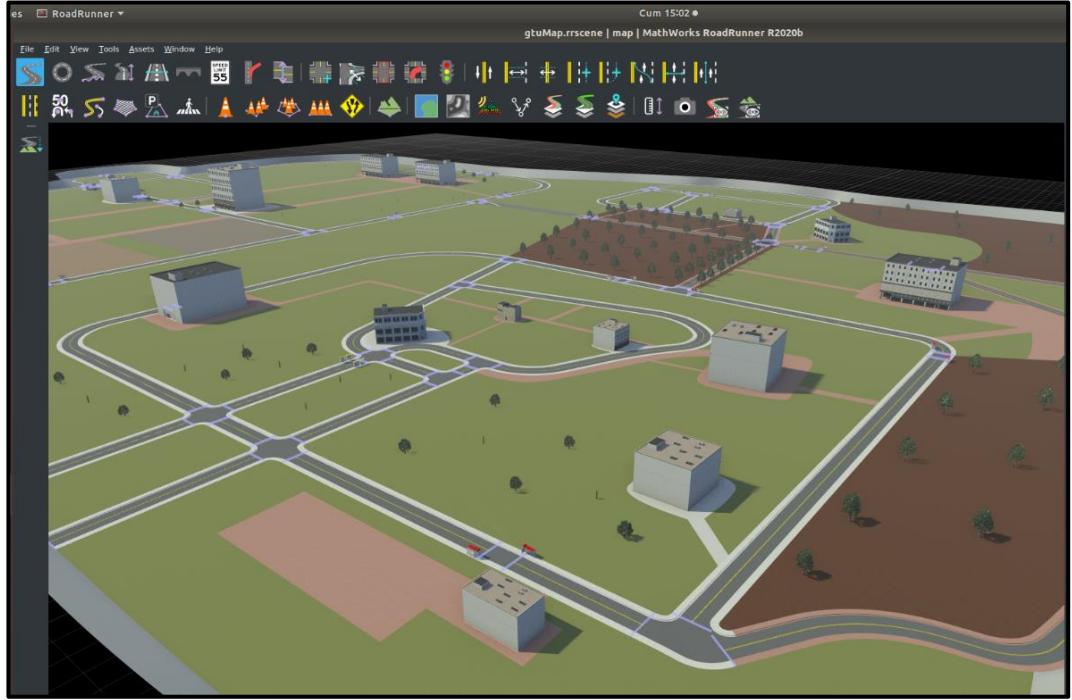


ŞEKİL 4.2 RoadRunner Harita Tasarımı – Çevre Unsurları ve Duvar Varlıkları

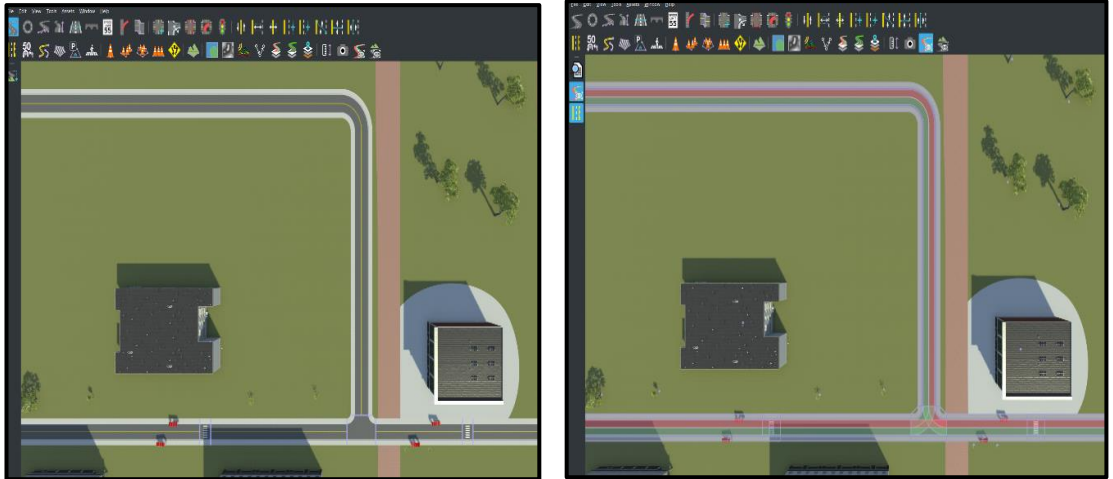
Harita tasarımının RoadRunner aşaması tamamlandıktan sonra (ŞEKİL 4.3 ve ŞEKİL 4.4) OpenDrive çıktısı alınarak yolların ve kesişim noktalarının geçerliliği test edildi. Hatalı kesişim noktalarındaki trafik akışı düzenlenerek geçerli hale getirildi. (ŞEKİL 4.5) Son aşamada CARLA Simülatöre aktarmak için harita .xodr (OpenDrive), .fbl ve .xml formatlarında dışarı aktarıldı.



ŞEKİL 4.3 RoadRunner Harita Tasarımı – Kuş Bakışı



ŞEKİL 4.4 RoadRunner Harita Tasarımı – Üç Boyutlu



ŞEKİL 4.5 RoadRunner ile OpenDrive Geçerlilik Kontrolü

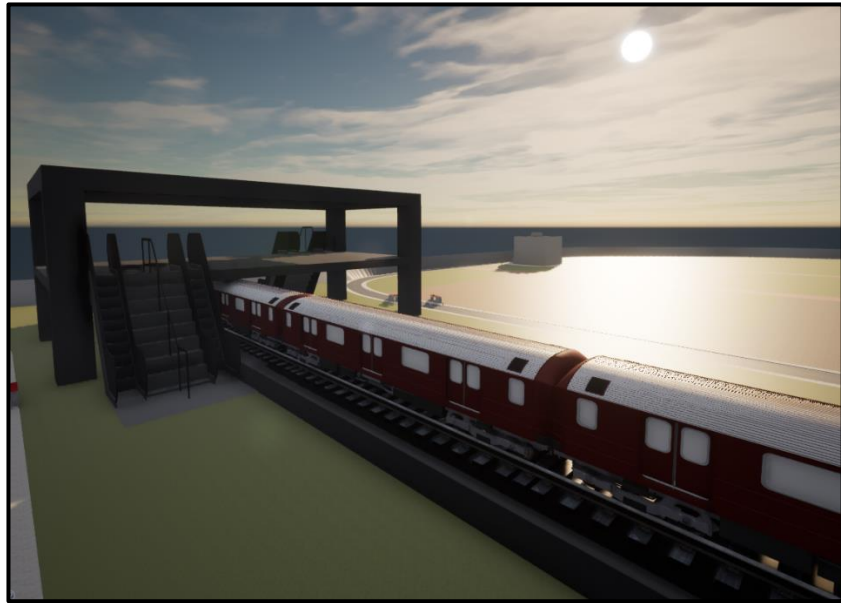
4.1.2. CARLA Simülâtör & Unreal Engine 4

CARLA, otonom sürüş sistemlerinin geliştirilmesi, eğitimi ve doğrulanmasını desteklemek için geliştirilmiş açık kaynak kod bir yazılımdır. [6] CARLA ortam simülasyonunu gerçekleştirebilmek için arka planda Unreal Engine 4 oyun motorundan

yararlanmaktadır. CARLA otonom sürüş için oluşturulan ve serbestçe kullanılabilen açık dijital varlıklar sağlar.[7] Projede bu varlıklar arasından araç ve yayalar kullanıldı.

Aynı zamanda simülasyon platformu, sensör paketlerini, çevresel koşulları, tüm statik ve dinamik aktörlerin tam kontrolünü, harita üretimini ve çok daha fazlasını destekler. [6] Bu özellikler içinden projenin geliştirilmesi sırasında hava durumunun belirlenmesinden, otonom aracın konum verisinin edinilmesinden, yaya ve araç dinamiğinin kontrolünden, tasarım haritanın simülatöre aktarımından ve yol noktası (waypoint) kullanımından yararlanıldı. Simülasyon ortamındaki otonom aracın sürüşü için haritada yer alan yol noktalarına erişildi ve aracın bu yol noktalarının takibi sağlandı, aynı zamanda fren, gaz ve direksiyon kontrollerinden yararlanıldı. Araç için sabit bir başlangıç ve varış noktaları belirlendi ve sürüş bu noktalar arasında gerçekleştirildi. Araç rotada sadece duraklarda duraklama yaptırıldı.

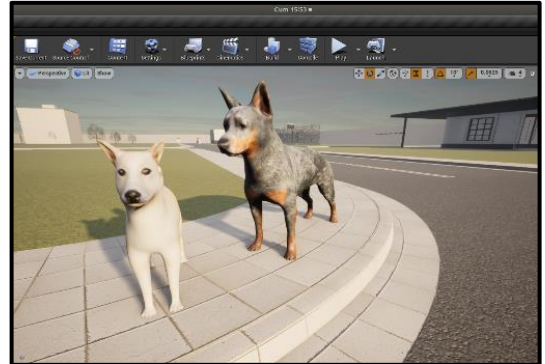
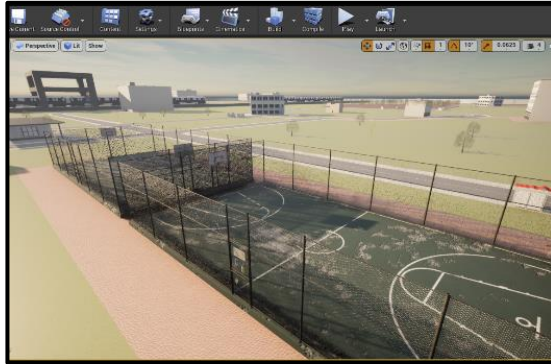
Simülasyon ortamını detaylandırmak için Unreal Engine 4 üzerinde tren istasyonu ve çeşitli çevre unsurlarının tasarlanmasında küp ve silindir varlıklarından yararlanıldı; tasarımlara doku ve renk eklemek için birçok materyaller geliştirildi. Tren istasyonu merdiveni, tren (ŞEKİL 4.6), güvenlik binası, güvenlik kapısı (ŞEKİL 4.7), basket sahası ve çeşitli hayvan dostlarımız (ŞEKİL 4.8) için .fbx formatında varlık tasarımları bulundu ve Unreal Engine 4 üzerinde birleştirilerek gerekli alanlarda konumlandırıldı.



ŞEKİL 4.6 CarlaUE4 Varlıklar – Marmaray Tren İstasyon



ŞEKİL 4.7 CarlaUE4 Varlıklar – Güvenlik Binası ve Kontrol Kapıları



ŞEKİL 4.8 CarlaUE4 Varlıklar – Basketbol Sahaları ve Hayvan Dostlarımız

Harita tasarımı .xodr ve .fbx formatında (campus.xodr ve campus.fbx) dışa aktarılıp, haritada kullanılan assetlerle birlikte CARLA kaynak dosyası içindeki import klasörüne konumlandırıldı. Aktarılan dosyalar işlenerek CarlaUE4 editörde “campus” isimli yeni bir level elde edildi. Level üzerinde navigasyonda hata alınmaması için; yayaların belirmesi istenen kaldırım isimlendirmesi sidewalk_campus olarak değiştirildi. Carla Exporter çalıştırılarak .obj dosyası (campus.obj) oluşturuldu. Sonrasında campus.obj ve campus.xodr dosyaları RecastBuilder’da verildi ve yaya navigasyonu için kullanılacak .bin (campus.bin) dosyası elde edildi.

4.2. AÇ GÖZLÜ ALGORİTMA

Proje danışmanı ile yapılan görüşmelerde BIL495 kapsamında araç için basit bir dinamik rota algoritması tasarlanmasına karar verilmişti. BIL496 kapsamında ise “Greedy Algorithm (Aç Gözlü Yaklaşım)”, “GRASP Greedy Randomized Adaptive Search (Aç Gözlü Randomize Uyarlanabilir Arama Prosedürü)”, “Tabu Search (Tabu Arama)” ve “Ant Colony Optimization (Karıncı Kolonisi Optimizasyonu)” olmak üzere 4 adet optimizasyon algoritmasının uygulanması hedeflenmişti, fakat CARLA simülatör hazır bir yapı olduğu için geliştirilen optimizasyon algoritmalarının ve threadlerin yapısının ortama entegrasyonunda çok fazla sorun meydana geldi, çözüm tahmin edilenden daha uzun bir sürede sağlanabildi. Bu sebeple rota algoritmasının optimizasyonunda sadece “Greedy Algorithm (Aç Gözlü Yaklaşım)” uygulanabildi.

Girdi dosyalarından okunan durak ve yolcu bilgileri kaydedildi. Bu veriler doğrultusunda CARLA simülatörde belirtilen duraklarda belirtilen zamanlarda farklı fiziksel özelliklerde yolcular oluşturuldu.

Araçlar için “carla” içinde yer alan “blueprint” kütüphanesinden bir araç modeli seçildi ve simülasyon ortamında 4 adet farklı renkte araç oluşturuldu. Aracın sürüşü ise “carla” kütüphanesine ait kontrol (control) yapısı ve yol noktası (waypoint) kullanımı ile gerçekleştirildi. Rotanın hesaplanmasında ise 3 farklı aşamada aç gözlü (greedy) yaklaşımdan yararlanıldı.

4.2.1. Rotadaki Araçların Yeni Rotaya İlerleme Kararı

Rotada seyreden bir araç her duraklamada yeni varış durağını belirler. Bu kararda etkili olan parametreler ise ikiye ayrılır. Araçta yolcu varsa öncelik içlerinden en yakın duraktakini bırakmaktır. Eğer araçta yolcu yoksa en yakın mesafedeki ve bekleyen yolcu bulunan durak seçilir. İki kararda da alınan yol mesafesi azaltıldığı ve yolcu konforu artırıldığı için amaç fonksiyonu değeri azalır ki bu istenen bir sonuçtur.

4.2.2. Rotadaki Aracın Sıradaki Yolu Seçme Kararı

Rotadaki bir araç duraklama yaptığı durakta öncelikle inmesi gereken yolcu/yolcular varsa bu aşamayı gerçekleştirir. Sonrasında kapasite kontrolü yapılır; yer varsa araçtaki yolcuların incekleri duraklar da göz önüne alarak en uygun yolcular seçilir.

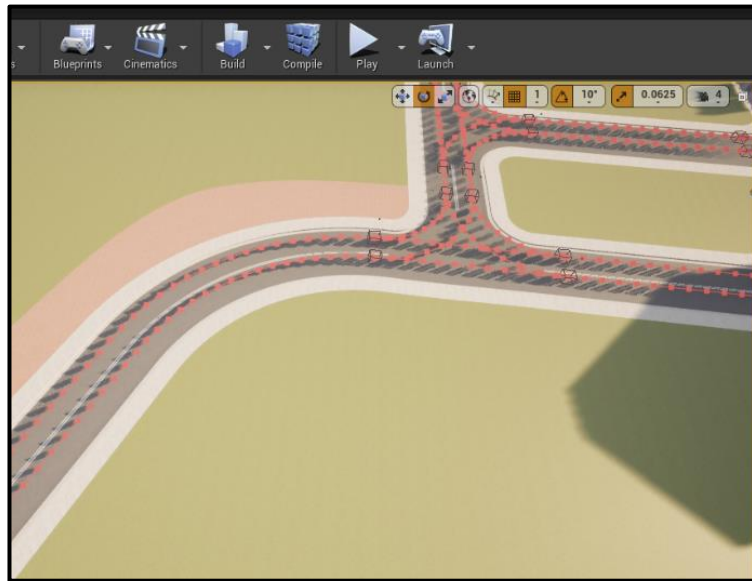
4.2.3. Yeni Otonom Aracın Rotaya Başlaması

4 adet otonom araç aynı anda rotaya çıkmamaktadır. Verimlilik elde etmek için araçlar gereksinime göre sırayla rotaya çıkmaktadır. Bu noktada rotada aracın 2 durak boyunca dolu kapasiteyle seyretmesi yeni aracın rotaya başlaması karar durumudur.

4.3. KONTROL VE YOL NOKTASI KULLANIMI

Yol noktası (waypoint) yapısı OpenDrive dosyasında yer alan bilgileri saklayan yapılardır. Harita üzerinde eşit aralıklarla yer alır ve tüm yollar için erişilebilirdir. Yol uzunluğu arttıkça waypoint sayısı da artmaktadır.

Kullanım için öncelikle haritaya ait tüm yol noktalarına (ŞEKİL 4.9) erişildi ve listede saklandı. Sonrasında sürüşün gerçekleşeceği yolların road_id ve lane_id değerlerine göre waypointler içinden filtreleme yapılarak gerekli yol noktası değerlerine ulaşıldı. Rota üzerindeki tüm yollarda bu yol yol noktası bilgileri sırayla algoritmada işlendi ve aracın bu noktaları takibi sağlanarak sürüş gerçekleştirildi. Yol üzerinde aracın hız kazanması için “kontrol (control)” sınıfına ait “gaz (throttle)” değişkenine 1 atandı; eğer yol üzerinde bulunan durakta duraklama gerekliyse konum bilgisi kontrolü yapılarak “kontrol (control)” sınıfına ait “fren (break)” değişkenine 1 atandı.



ŞEKİL 4.9 CarlaUE4 Üzerinde Yol Noktalarının Gösterimi

4.4. THREADLER

Program içerisinde “time_counter” ve “spawning” isimli iki thread kullanıldı; bir tanesi simülasyon boyunca her 30 saniyede bir sayaç değerini arttırmaktadır. İkincisi ise alınan yolcu bilgisine göre belirli zamanlarda ve konumlarda yolcuların oluşmasını simülasyon boyunca her yeni sayaç değerinde tüm yolcuların kontrollünü yaparak sağlamaktadır.

İki thread simülasyon boyunca senkronize çalışmaktadır, bunun için 2 tane mutex kullanılmıştır. ŞEKİL 3.6 da thread senkronizasyonu için kullanılan yapının sözde kod (pseudocode) versiyonunu görebilirsiniz.

<pre>procedure timeCounter(): time_counter = 0 for i in range(simülasyon_süresi): lock1'i yakala/kilitle time_counter = time_counter + 1 sleep(30) lock2'yi serbest bırak/aç print “süre doldu”</pre>	<pre>procedure spawning(world): for i in range(yolcu_sayısı): lock2'i yakala/kilitle for i in range(yolcu_sayısı) if passengerList[i]'nin oluşma zamanı time_counter'a eşitse: passengerList[i] 'i için konum bilgisini al passengerList[i] 'i oluştur lock1.release()</pre>
--	---

ŞEKİL 4.10 Thread Senkronizasyonu – Sözde Kod

```
campusData.txt • stopCoordinates.txt
home > ezgi > Documents > kod > campusData.txt
1 4
2 1 stop_101 1 stop_28 15
3 2 stop_101 1 stop_28 15
4 3 stop_28 2 stop_101 15
5 4 stop_28 2 stop_101 15
6 5 stop_28 2 stop_101 15
7 6 stop_28 2 stop_101 15
8 7 stop_0 3 stop_101 15
9 8 stop_0 3 stop_101 15
10 9 stop_0 3 stop_101 15
11 10 stop_2 3 stop_101 15
12 11 stop_2 3 stop_101 15
13 12 stop_2 3 stop_101 15
14 13 stop_2 3 stop_101 15
15 14 stop_2 3 stop_101 15
16 15 stop_2 3 stop_101 15
17 16 stop_2 3 stop_101 15
18 17 stop_2 3 stop_101 15

ezgi@ezgi:~/Documents/kod$ python3 e
WARNING: cannot parse georeference:
counter: 1
create passenger id 1
create passenger id 2
counter: 2
create passenger id 3
create passenger id 4
create passenger id 5
create passenger id 6
counter: 3
create passenger id 7
create passenger id 8
create passenger id 9
create passenger id 10
create passenger id 11
create passenger id 12
create passenger id 13
create passenger id 14
create passenger id 15
create passenger id 16
create passenger id 17
counter: 4
counter: 5
counter: 6
```

ŞEKİL 4.11 Yolcu Bilgisi ve Thread Senkronizasyonu ile Oluşturulması

5. SONUÇ

Bu bölümde geliştirilen simülasyon ortamını ve rota algoritmalarının sayısal sonuçlarına dair uygulamalara yer verilmiştir.

5.1. SİMÜLASYON VE SÜRÜŞ SONUCU

Kullanılan Sistem

- Ubuntu 18.04 ve üzeri işletim sistemi
- Python 3.6.9 derleyicisi
- En az 100GB disk alanı
- 16 GB RAM
- 1.8Ghz 6GB GPU – GeForce GTX 1060 6GB
- 3.20 – 4.60 GHz CPU – intel i7-7700

Bu sistemde tasarlanan “Aç Gözlü (Greedy) Yaklaşım” farklı test dosyalarıyla çalıştırılarak her araç için taşınan yolcu ve rota mesafesi hesaplanmıştır. Taşınamayan yolcu sayısı ise en son sütunda yer almaktadır. (TABLO 5.1)

Algoritma geliştirilirken özellikle “yolcu karar” aşamasında istenen sonuç elde edilemedi. Yolcular duraktan alınırken sadece aracın kapasitesi kriter alındı, bu sistemin istenmemesinin sebebi ise taşınan fakat dilediği noktaya varamayan yolcuların oluşmasının mümkün olmasıdır. Fakat son aşamada daha gelişmiş bir algoritma tasarlanamamıştır. Yine aşamalardan bir başkası olan “sonraki durak kararı” nda ise araç dolu kapasitede iken taşıdığı yolculara göre karar vermek yerine algoritmadaki eksiklerden dolayı en yakın mesafedeki konuma ilerlediği için istenen noktaya ulaşamamış yolcular olabilmektedir. Bu durumun da hedeflenmeyen sonuçlara yol açması mümkündür.

TABLO 5.1 Aç Gözlü Yaklaşım Sonuçları

Test	Taşınan Yolcu Araç 1	Taşınan Yolcu Araç 2	Taşınan Yolcu Araç 3	Taşınan Yolcu Araç 4	Rota Uzunluğu Araç 1	Rota Uzunluğu Araç 2	Rota Uzunluğu Araç 3	Rota Uzunluğu Araç 4	Taşınamayan Yolcu
Test 1	10	2	0	0	753 m	235 m	0 m	0 m	0
Test 2	15	12	4	0	1087 m	753 m	670	0 m	0
Test3	28	24	10	2	1267 m	952 m	753 m	235 m	0
Test 4	28	24	12	10	1502 m	952 m	783 m	753 m	4

5.2. HARİTA TASARIM SONUCU

Simülasyon ortamı için GTÜ kampüsünün tasarımı başarılı sonuçlanmıştır. Ortamı tanıyan 20 kişiden belirlenen 5 kategoride 10 üzerinden puan vermeleri istendi. Tablonun en alt satırında yer aldığı gibi tüm kategorilerde ortalama puan 9 üstü oldu. (TABLO 5.2)

TABLO 5.2 Harita Tasarımının Kategorilere Göre Puanlama Sonuçları

Kişi	Genel Görünüm	Yollar ve Alt Geçit	Çevre Düzenlemesi	Tren İstasyonu	Orman Arazisi
Kişi 1	10	10	10	9	10
Kişi 2	10	10	9	8	10
Kişi 3	10	10	10	10	10
Kişi 4	9	10	9	9	10
Kişi 5	10	10	10	10	10
Kişi 6	10	10	10	10	10
Kişi 7	10	8	9	9	10
Kişi 8	10	10	10	8	9
Kişi 9	9	10	10	8	9
Kişi 10	10	10	9	10	8
Kişi 11	9	9	10	8	10
Kişi 12	10	10	10	10	10
Kişi 13	10	10	10	10	10
Kişi 14	9	10	9	7	7
Kişi 15	10	10	10	9	9
Kişi 16	10	10	9	9	8
Kişi 17	10	10	10	10	10
Kişi 18	9	10	7	9	8
Kişi 19	10	10	10	10	10
Kişi 20	10	10	10	10	10
	9,75	9,85	9,55	9,15	9,4

5.3. PROJE GELİŞTİRİLİRKEN KARŞILAŞILAN ZORLUKLAR

- Yolcuların zamana bağlı oluşması için simülasyon süresince sürekli olarak çalışması istenen bir yapıya ihtiyaç duyuldu. Bunun için de thread tercih edildi. Bu aşamada daha önce Python’ da thread ile çalışılmamış olması ve CARLA ortamında thread kullanımına dair fazla kaynak bulunmamasından dolayı tahmin edilenden daha uzun sürede çözüme ulaşıldı.
- Hedeflenenin aksine “GRASP Greedy Randomized Adaptive Search (Aç Gözlü Randomize Uyarlanabilir Arama Prosedürü)”, “Tabu Search (Tabu Arama)” ve “Ant Colony Optimization (Karıncı Kolonisi Optimizasyonu)” olmak üzere 3 adet meta sezgisel optimizasyon algoritmasının uygulanma hedefi gerçekleştirilemedi. Sebebi ise sayısal verilerin görsel aktarımında simülatör ihtiyacı olmasıydı. CARLA simülatör hazır bir yapı olduğu için geliştirilmek istenen kompleks optimizasyon algoritmalarının entegrasyonunda çok fazla sorun meydana geldi. CARLA üzerinde anlık kararlara dayanan bir mekanizma yer alırken meta sezgisel algoritmalar birçok deneme sonucunda en iyiye ulaşmayı hedefleyen algoritmalarlardır. Bu uyumsuzlıktan dolayı uygun algoritmalar geliştirilemedi, çözüm elde edilemedi.

6. KAYNAKÇA

- [1] [Nestor Subiron, CARLA: Open-source Simulator for Autonomous Driving Research](#)
- [2] [Dosovitskiy, Ros, Codevilla, L'opez & Koltun, , CARLA: An Open Urban Driving Simulator , 2017](#)
- [3] Kızıldaş, Mehmet Çağrı, A Review of Characteristics and Existing Case and Future Position of Autonomous Vehicles on the Context of Effects over Transportation and Traffic, DÜMF Mühendislik Dergisi, (2020): 1251-1259, 2020
- [4] [RoadRunner Hakkında Bilgi](#)
- [5] [RoadRunner Editör Kullanımı İçin Dökümantasyon](#)
- [6] [CARLA Simulator Hakkında Bilgi](#)
- [7] [CARLA Kurulum ve Sistem Gereksinimleri](#)
- [8] [CARLA Client - Server Dökümantasyonu](#)
- [9] [CARLA Python API Dökümantasyonu](#)
- [10] [CARLA Simülatöre Yeni Harita Ekleme Dökümantasyonu](#)
- [11] [CARLA Simülatöre Yeni Harita Ekleme İçin Veri İşleme](#)
- [12] [CARLA Blueprint Kütüphanesi Hakkında Dökümantasyon](#)
- [13] [CARLA Simülatör Aktörler Hakkında Dökümantasyon](#)
- [14] [Waypoint Kullanımı](#)
- [15] [Oyola, Lokketangen GRASP-ASP: An Algorithm for the CVRP with the Route Balancing, August 2014](#)
- [16] [Unreal Engine 4 Ortama Varlık \(Asset\) Ekleme](#)
- [17] [Unreal Engine 4 Varlık \(Asset\) Düzenleme](#)
- [18] [Unreal Engine 4 Level Editor](#)
- [19] [NetworkX Kütüphanesi](#)
- [20] [Carla Multi-Client](#)